On the robustness of conjugate-gradient methods and quasi-Newton methods

Mikael FALLGREN, werty@kth.se

Master Thesis, 5B1022
Department of Mathematics
Royal Institute of Technology
7/9 - 2006

Abstract

On a quadratic problem the conjugate-gradient method and the quasi-Newton method are equivalent, if exact line search is applied. The aim of this master thesis is to investigate whether there are noticeable differences between the methods when solving a nonquadratic problem, when performing inexact line search or when both alternatives are applied. In this research three convex objective functions with variable size are studied. The line search methods applied are exact line search, backtracking line search and Wolfe line search. The results indicate that the best method seem to depend on which problem that is to be solved. Within the conjugate-gradient method it tend to be preferable to use the Polak-Ribière method or alternatively the Hestenes-Stiefel method. The quasi-Newton methods to be preferred are the BFGS method and the DFP method, which perform similarly. If the problem has a convex objective function, then the BFGS method probably is to favor. This is due to the known theoretical result of global convergence for a specific inexact line search method. If a problem is going to be solved repeatedly within some application, it might be of interest to perform a similar comparison of the methods.

Key words. Unconstrained optimization, conjugate-gradient method, quasi-Newton method, line search method.

Contents

1. Introduction

The conjugate-gradient method (CG) and the quasi-Newton methods (QN) are equivalent for a quadratic objective function when exact line search is used. This was proved by Nazareth [20], using Dixon [7]. The aim of this research is to investigate differences between CG and QN, when a perturbation is added to the equivalent situation stated above. One variation is that the objective function is close to quadratic, while exact line search is used. The other variation is that the objective function is quadratic, while inexact line search is used. It is also possible to include both variations at the same time, such that the objective function is not quadratic and the line search is not exact.

The CG and QN are methods that only use the first derivative of the function. They are therefore often used in applications when only the first derivative is known, or when higher derivatives are very expensive to calculate. In some applications the optimal solution might not be of particular interest, instead the interest might be the behavior of a particular method. The methods can be used in a great variety of fields, some examples from different areas are presented below. In electromagnetics, problems for distributed parameter estimation can be solved, as in [13], where a constructed update matrix for QN is taking advantage of the special structure of the problem. Optical tomography is another area where these techniques are used [16]. These two methods can also be used in programs of neural network, see, e.g., [6] and [27]. Finally, the methods can be used in radiation therapy optimization [4], from which this study has evolved.

The formulation of the unconstrained optimization problem to be investigated is

$$\underset{x\in I\!R^n}{\text{minimize}} \quad f(x), \tag{1.1}$$

where f is a smooth, convex function with its gradient $\nabla f(x)$ available. This study originates from the problem where $f(x)$ is a quadratic function

$$f(x) = \frac{1}{2}x^T Hx + c^T x, \tag{1.2}$$

where the Hessian H is a symmetric and positive definite $n \times n$ matrix and c is a vector of length n. With a positive definite Hessian matrix H, the objective function $f(x)$ is convex.

To solve the stated unconstrained optimization problem (1.1) with the objective function (1.2) is equivalent to solving a linear system $Ax = b$, where $H = A$, $c = -b$. There are several strategies on how to solve a linear system of equations with symmetric and positive definite matrix. Some well known strategies are Cholesky factorization, Householder rotation and Lanczos method. These methods are briefly described in Appendix A. The Lanczos method is mathematically equivalent to the conjugate-gradient method applied on an unconstrained optimization problem (1.1) with the objective function (1.2). This relation is presented in some more detail in section 2.4.

Within this research, the unconstrained optimization problem (1.1) will be solved by a strategy which track the optimal x-value by starting with an initial x-value x_0 and thereafter update the x by performing iterations of the form

$$x_{k+1} = x_k + \alpha_k p_k, \tag{1.3}$$

where p_k is a search direction and α_k is a stepsize. The introduced index k represent the number of iterations proceeded by the method. To be able to take the step towards a new point x_{k+1}, the search direction p_k and the stepsize α_k have to be found. Methods to find a search direction p_k and a stepsize α_k are presented in sections 2, 3 and 4. The methods are implemented so that p_k is a descent direction, i.e. $p_k^T \nabla f(x_k) < 0$ is fulfilled [21]. Note that the update of x_{k+1} in (1.3) makes the expression $x_{k+1} = \alpha_k p_k + \alpha_{k-1} p_{k-1} + ... + \alpha_0 p_0 + x_0$ valid. The point x_{k+1} therefore lies in the subspace spanned by $\{x_0, p_0, p_1, ..., p_k\}$. If applying the conjugate-gradient method and exact line search in exact arithmetic on the convex quadratic function (1.2), then the point $x_{k+1} \in \{x_0\} \cup \{p_i\}_{i=0}^{k}$ minimizes the quadratic function over this subspace [19, p.385].

There have been numerous studies on CG methods, QN methods and on variations between the methods. Within the CG methods, the studies in general seem to suggest that it is preferable to use the Polak-Ribière method, see, e.g., [18, p.254] and [22, p.122]. Within the QN methods, it is in general believed that the most effective update method within the Broyden family is the BFGS method [11, p.119]. If comparing CG with QN the general opinion seems to be that the QN is to be preferred compared to CG, since conjugate-gradient methods are less efficient and less robust compared to quasi-Newton methods. However, with no matrix operations and relative little storage CG is a good method for large problems [8, pp.84-85].

The outline of this report is as follows. In section 2 a background on the conjugate-gradient method is given. This section is divided into four subsections containing formulation and criteria, update methods of β_{k-1}, algorithm and alternative reformulations. Section 3 gives background information on quasi-Newton methods. This section is categorized into five subsections containing formulation and criteria, update methods of U_k, algorithm, self-scaling quasi-Newton methods and adapt for large problems. In section 4 some line search methods are presented, which are used to obtain the stepsize. In four subsections exact line search, backtracking line search, Armijo's rule and Wolfe conditions are presented. Section 5 contains a brief description of the computing environment. In section 6 the results of this study are presented, divided into three subsections containing Testproblem 1, Testproblem 2 and Testproblem 3. Section 7 contains a discussion and finally a conclusion is presented in section 8.

2. The conjugate-gradient method

The conjugate-gradient method (CG) is to be described in this section. In a paper by Hestenes and Stiefel [14], the conjugate-gradient method was originally presented [19, p.407]. The conjugate-gradient method is a low storage algorithm that generates search directions to track the optimum of an unconstrained optimization

problem (1.1) [19, p.383]. It is designed to have a faster convergence then steepest descent while avoiding evaluation and storage of matrices [18, p.238].

For a quadratic function (1.2) where the Hessian matrix H has m distinct eigenvalues, the conjugate-gradient method will find the solution in at most m iterations [22, p.114]. With m distinct clusters of eigenvalues, the conjugate-gradient method will approximately solve the problem in m iterations [22, p.116].

2.1. Formulation and criteria

The formulation of CG and some important criteria are to be presented in this subsection. Let the gradient evaluated at a point x_k be denoted by

$$g_k = g(x_k) = \nabla f(x_k).$$

If the gradient g_k is equal to zero, the global optimum is found for the point x_k. An unconstrained optimization problem with the objective function (1.2) evaluated at a point x_k, has the gradient given by $g_k = Hx_k + c$. The negative gradient $-g_k$ can be represented as a residual, which the CG method decrease during the iterations. A global optimum is found when the residual is equal to zero. Note that CG's gradients are mutually orthogonal, $g_k^T g_i = 0, \forall\, i = 0, ..., k-1$ [22, p.108].

The first search direction p_0 is performed along the steepest descent direction $-g_0$. Thereafter the search direction p_k is a linear combination of the steepest descent $-g_k$ and the previous search direction p_{k-1} scaled with a factor β_{k-1},

$$p_k = -g_k + \beta_{k-1} p_{k-1}.$$

The scalar β_{k-1} is determined with the requirement that p_{k-1} and p_k must be conjugate with respect to the Hessian H for problem (1.2) [22, p.107]. There are several methods on how to compute the scalar β, these are discussed in more detail in section 2.2. An obtained nonzero gradient g_k is linearly independent of the previous gradient directions and of the previous search directions. The gradient g_k is orthogonal to the subspace generated by $\{p_0, \ldots, p_{k-1}\}$ [18, pp.243-244]. Hence, the gradients g_j satisfies the condition

$$g_j^T p_i = 0, \quad \text{for } i < j,$$

where $i = 0, ..., j-1$ and $j = 1, ..., k,$. The condition is known as the *expanding subspace theorem* [18, p.242]. For a convex quadratic problem (1.2) when exact line search is applied the conjugate-gradient method generates search direction vectors $\{p_i\}$ that are mutually conjugate with respect to the Hessian matrix H [19, p.383]. When the vectors $\{p_j\}$ are mutually conjugate with respect to the Hessian matrix H then

$$p_i^T H p_j = 0 \quad \text{if } i \neq j, \tag{2.1}$$

for $i = 0, \ldots, k$ and $j = 0, \ldots, k$, which is known as the *conjugacy condition*. The conjugacy condition (2.1) is equivalent with the orthogonality condition ($g_{i+1} -$

$g_i)^T p_j = 0$ for $i \neq j$ [11, p.145]. Satisfying (2.1) implies linearly independent vectors [22, p.102]. It is from the conjugacy criteria the conjugate-gradient method has got its name [19, p.383].

By successively minimizing along each direction of the conjugate set, the objective function (1.2) will be minimized in at most n steps if exact arithmetic is used [11, p.147]. When conjugate-gradient methods are applied to nonquadratic problems, they usually do not terminate within n steps. Hence, the algorithm should continue with its search until a termination criteria is met. Alternatively CG can be interrupted after a certain number of iterations and restarted [18, p.252].

2.2. Update methods of β_{k-1}

Three update methods of β_{k-1} are to be presented in this subsection. As previously mentioned, there are different methods on how to choose the scalar β_{k-1}. Below, the three best known formulas for β_{k-1} are presented:

- Fletcher-Reeves method (FR)

$$\beta_{k-1} = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}, \tag{2.2}$$

- Hestenes-Stiefel method (HS)

$$\beta_{k-1} = \frac{(g_k - g_{k-1})^T g_k}{(g_k - g_{k-1})^T p_{k-1}}, \tag{2.3}$$

- Polak-Ribière method (PR)

$$\beta_{k-1} = \frac{(g_k - g_{k-1})^T g_k}{g_{k-1}^T g_{k-1}}. \tag{2.4}$$

These methods are equivalent when applied to a quadratic objective function (1.1), but not when applied to a general nonlinear function. The FR method was originally discussed in a paper [9] by Fletcher and Reeves. If the FR formula is used, the method will converge for appropriate assumptions. For PR and HS, there are constructed examples where these two formulations have great difficulties. However, computational experiments suggest that both PR and HS perform better than FR [19, p.399]. In general the Polak-Ribière method seems to be preferable compared to other methods, according to Luenberger [18, p.254]. According to Nocedal, no β_{k-1} update has proved to be significantly more efficient than the Polak-Ribière update [22, p.122].

2.3. Algorithm

Previously in this section formulas have been presented on how to calculate various important parameters within the conjugate-gradient method, at a certain iteration k. To summarize the main parts of the method, one version is presented on the following page as an algorithm.

Algorithm 2.1. *The conjugate-gradient algorithm*

$k \leftarrow 0$; $x_k \leftarrow$ initial point; $g_k \leftarrow \nabla f(x_k)$;
while $\|g_k\|_2 \neq 0$ **then**
  if $k > 0$ **then**
    $\beta_{k-1} \leftarrow$ update method: (2.2), (2.3) or (2.4);
    $p_k \leftarrow -g_k + \beta_{k-1} p_{k-1}$;
  else
    $p_k \leftarrow -g_k$;
  end
  $\alpha_k \leftarrow$ line search method: (4.1), (4.3), Algorithm 4.1, 4.2 or 4.3;
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$; $g_{k+1} \leftarrow \nabla f(x_{k+1})$;
  $k \leftarrow k + 1$;
end

In this algorithm all variables are written with subscripts. Only the current values of the variables need to be saved, with two exceptions. The exceptions are the previous search direction and the previous gradient direction, since they are needed in the update of β.

2.4. Alternative reformulations

Some alternative reformulations will be presented in this subsection. For some applications it might be advantageous to reformulate the optimization problem. Within this subsection the optimization problem (1.1) with the objective function (1.2) is going to be considered. Let the initial x-value be x_0 and let x^* denote the unknown optimal solution to the problem, i.e. $x^* = H^{-1}c$. The objective function (1.2) can then be reformulated as

$$f(x) = \frac{1}{2}x^T H x + c^T x = \frac{1}{2}(x - x^*)^T H (x - x^*) - \frac{1}{2}x^{*T} H x^*. \tag{2.5}$$

One reformulation alternative is to express the vector $x_0 - x^*$ in an eigenvector expansion as

$$x_0 - x^* = \zeta_1 \, \mathbf{e}_1 + \zeta_2 \, \mathbf{e}_2 + ... + \zeta_n \, \mathbf{e}_n,$$

where $\mathbf{e}_i$ is normalized eigenvectors of the Hessian H. Then

$$H(x_0 - x^*) = \lambda_1 \zeta_1 \, \mathbf{e}_1 + \lambda_2 \zeta_2 \, \mathbf{e}_2 + ... + \lambda_n \zeta_n \, \mathbf{e}_n,$$

where λ_i are the corresponding eigenvalues of the Hessian H. The constant term $\frac{1}{2}x^{*T} H x^*$ of the objective function (2.5) can be ignored by defining

$$\tilde{f}(x) = f(x) + \frac{1}{2}x^{*T} H x^* = \frac{1}{2}(x - x^*)^T H (x - x^*), \tag{2.6}$$

and then study the translated objective function $\tilde{f}(x)$. Note that the eigenvectors of H are mutually orthogonal and therefore

$$\tilde{f}(x_0) = \frac{1}{2}(x_0 - x^*)^T H (x_0 - x^*) = \frac{1}{2}\sum_{i=1}^{n} \lambda_i \zeta_i^2,$$

where λ_i are the corresponding eigenvalues and $\zeta_i \mathbf{e}_i$ are the corresponding eigenvectors of the Hessian H [18, pp.242-247].

Another reformulation alternative is performed on the CG method. Consider the formulation of the translated objective function (2.6). Define $\xi_0 = x_0 - x^*$, which gives the initial gradient $g_0 = H\xi_0$. The conjugate-gradient method at iteration $k+1$ can then be written as

$$\begin{array}{ll} \underset{x \in I\!R^n}{\text{minimize}} & \frac{1}{2}\xi^T H \xi \\ \text{subject to} & \xi = \xi_0 - G_k \alpha, \end{array} \tag{2.7}$$

where $G_k = (g_0, g_1, \ldots, g_k)$. The obtained optimal solution is ξ_{k+1} and α_{k+1}. The gradient is given by $g_{k+1} = H\xi_{k+1}$, or alternatively

$$g_{k+1} = g_0 - \sum_{h=0}^{k} (H g_h \alpha_h). \tag{2.8}$$

The gradient formulation (2.8) show that the gradient g_{k+1} is contained in the Krylov subspace $\mathcal{K}_{k+1}(H, g_0)$, defined as

$$\mathcal{K}_{k+1}(H, g_0) = span\{g_0, Hg_0, \ldots, H^k g_0\}, \tag{2.9}$$

see, e.g., [29, p.152]. Elimination of ξ in (2.7) gives the optimal solution

$$G_k^T H G_k \alpha = -G_k^T H \xi_0 = -G_k^T g_0. \tag{2.10}$$

The matrix $G_k^T H G_k$ is tridiagonal, this is due to the fact that $g_i^T H g_j = 0$ for $i \geq j-2$, where $i = 2, \ldots, k$ and $j = 2, \ldots, i$. This claim is valid due to the orthogonality of the gradients

$$0 = g_i^T g_j = g_i^T (g_0 - \sum_{h=0}^{j-1} (H g_h \alpha_h)), \quad \text{for } i > j.$$

Thus each element outside the tridiagonal of $G_k^T H G_k$ equal zero. Now form the $L_k D L_k^T$ factorization of the tridiagonal matrix, which gives

$$L_k D_k L_k^T = G_k^T H G_k \;\Leftrightarrow\; D_k = L_k^{-1} G_k^T H G_k L_k^{-T}.$$

Define $P_k = -G_k L_k^{-T}$, where $P_k = (p_1, \ldots, p_k)$. This gives $D_k = P_k^T H P_k$, i.e. the directions in P_k are conjugate with respect to the Hessian matrix H. The equation $P_k L_k^T = -G_k$ at column k is $P_k L_k^T e_k = -g_k$, where L_k is a unit lower-bidiagonal matrix, which gives

$$p_k + l^{k,k-1} p_{k-1} = -g_k. \tag{2.11}$$

Multiplication with $p_{k-1}^T H$ in (2.11) and simplification gives

$$l^{k,k-1} = -\frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}.$$

Consequently the conjugate-gradient method is obtained, where the exact stepsize $\alpha_k = \alpha_k(k)$ and the $\beta_{k-1} = l_{k,k-1}$, which is Fletcher-Reeves updating method (2.2).

As mentioned in the introduction, the Lanczos method is a strategy for solving a linear system $Ax = b$ that is mathematically equivalent to the conjugate-gradient method. The equivalence is motivated below. Define an upper bidiagonal matrix

$$\Omega_k = \begin{pmatrix} 1 & -\beta_1 & 0 & \dots \\ 0 & \ddots & \ddots & 0 \\ \vdots & 0 & \ddots & -\beta_{k-1} \\ \vdots & \vdots & 0 & 1 \end{pmatrix}.$$

This gives the relation $G_k = -P_k\Omega_k$. The tridiagonal matrix $G_k^T H G_k = \Omega_k^T \Lambda_k \Omega_k$, where $\Lambda_k = diag(p_1^T H p_1, \dots, p_k^T H p_k)$. Define

$$Q_k = -G_k \Delta^{-1},$$

where $\Delta = diag(\sqrt{g_0^T g_0}, \dots, \sqrt{g_{k-1}^T g_{k-1}})$. The column vectors of the matrix Q_k then form an orthonormal basis for the Krylov subspace (2.9). The columns of Q_k are the Lanczos vectors, whose projection of the Hessian H is $\Delta^{-1}\Omega_k^T \Lambda_k \Omega_k \Delta^{-1}$ [12, pp.370-371]. The strategy of the Lanczos method is to produce a set of orthonormal vectors $\{q_j\}$ and then letting x_k minimize the problem over the $span\{q_1, \dots, q_k\}$. For some y_k it follows that $x_k = Q_k y_k$ [12, p.342], where

$$(Q_k^T H Q_k) y_k = -Q_k^T c.$$

Note the similarities between this equation and (2.10) with $g_0 = H(x_0 - x^*) = Hx_0 + HH^{-1}c = c$, for $x_0 = 0$. It is from this the Lanczos method is derived, see Appendix A3 for the Lanczos method.

3. Quasi-Newton methods

Quasi-Newton methods are iterative methods for solving an unconstrained minimization problem (1.1). The methods are based on building up curvature information during the iterations, while the descent method is running. It approximates the curvature of the objective function without forming the Hessian matrix itself [11, p.116]. This make the iterations of QN computationally more expensive, compared to the CG. However, stored information in the approximated Hessian might decrease the total number of iterations compared to the conjugate-gradient method.

3.1. Formulation and criteria

The formulation of QN is to be presented, as well as some important criteria. The search direction p_k is obtained by solving following linear system

$$B_k p_k = -\nabla f(x_k),$$

where B_k is a symmetric and positive definite matrix. Note that if $B_k = \nabla^2 f(x_k)$ then it would be Newton's method, while $B_k = I$ gives the steepest descent method.

During the iterations the matrix B_k is building up so it approximates the Hessian $\nabla^2 f(x_k)$, while the objective function is minimized [19, pp.339-340].

Another way to present the QN method is by approximating the inverse Hessian matrix formula instead, by letting $B_k = H_k^{-1}$ [11, p.122]. The aim is to avoid solving a linear equation system each iteration. Note that for linear system of equations, the inversion approach is much less stable [15, p.262]. The inverse formulation of the Hessian approximation matrix is therefore not used in this research.

The initial Hessian approximation B_0 is often set equal to the identity matrix, $B_0 = I$, if no further information is available [11, p.117]. Within this research, the choice of initial Hessian approximation is $B_0 = I$. The initial Hessian approximation can also be set to a multiple of the identity matrix $B_0 = \eta I$, where η is a constant. However, there is no known good general strategy on how to choose η. Another method is to start with $B_0 = I$ and after computing the first step, but before using the update U_k formula, the initial matrix is changed according to $B_0 \leftarrow \frac{y_k^T y_k}{y_k^T s_k} I$ [22, p.200].

The Hessian approximation matrix B_k is updated according to

$$B_{k+1} = B_k + U_k.$$

There are several strategies for updating the U_k matrix. Some of these are presented in section 3.2. A condition to define the matrix B_k is

$$B_k(x_k - x_{k-1}) = \nabla f(x_k) - \nabla f(x_{k-1}), \tag{3.1}$$

which is known as the *secant condition*. This condition is based on a generalization of $f''(x_k) \approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$, where the formula first is made multidimensional and then the Hessian term $\nabla^2 f(x_k)$ is replaced by an approximation term B_k. Two vector definitions follow below, these are introduced for simplicity and are used repeatedly further on,

$$\begin{cases} s_k = x_{k+1} - x_k = \alpha_k p_k, \\ y_k = \nabla f(x_{k+1}) - \nabla f(x_k). \end{cases} \tag{3.2}$$

Note that $y_k = H s_k$ for the quadratic problem (1.2). Combining the secant condition (3.1) with the two vector definitions (3.2) gives a simplified expression of the secant condition

$$B_{k+1} s_k = y_k, \tag{3.3}$$

see, e.g., [19, p.349]. This condition is required to hold for the new updated matrix B_{k+1} [11, p.118]. Note that it is only possible to fulfill the secant equation if

$$s_k^T y_k > 0, \tag{3.4}$$

which is known as the *curvature condition*. Infact, the secant condition always has a solution if the curvature condition is satisfied [22, pp.195-196].

3.2. Update methods of U_k

The update matrix U_k can be calculated by different methods. Infact, one is a whole class of formulas which will be used within this research. The method is given by

$$\begin{cases} U_k = -\frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \phi_k (s_k^T B_k s_k) v_k v_k^T, \\ v_k = \frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k}, \end{cases} \tag{3.5}$$

where ϕ_k is a scalar. This class is known as the Broyden class, see, e.g., [19, p.355] and [22, p.207], or as the one-parameter family of updates [11, p.119]. It is also referred to as the Broyden family in [18, p.270] and [8, p.56]. Further on in this report, this class of formulas will be referred to as the Broyden family. The restricted Broyden family is when $0 \leq \phi_k \leq 1$ [22, p.208], which also is known as the convex class [11, p.126].

Let x_0 be a starting point and B_0 an initial symmetric and positive definite Hessian approximation matrix. Take an update of the Broyden family which for all k satisfies that ϕ_k is larger than the critical value for which the B_{k+1} update turns indefinte. Apply it on a strongly convex quadratic function, with exact line search in exact arithmetic. Then following statements are valid:

- At most n iterations are used to converge to the solution.
- For all previous search directions, the secant condition is satisfied: $B_k s_j = y_j, \quad j = k-1, ..., 1.$
- With starting matrix $B_0 = I$, the iterates are identical to those generated by the conjugate-gradient method. Note that the search directions are conjugate, $s_i^T H s_j = 0$ for $i \neq j$, where H is the Hessian of the quadratic function (1.2).
- $B_{n+1} = H$, if n iterations are performed [22, p.210].

Note that if an exact line search is used in every iteration, on a twice-continuously differentiable function, all the methods of the Broyden family generate identical points, as long as each sequence of $\{\alpha_i\}_{i=0}^k$ and $\{B_i\}_{i=0}^k$ is well defined [11, p.120].

Below, three different update methods U_k within the Broyden family are presented. It is the symmetric rank one update method, the BFGS method and the DFP method.

The *symmetric rank one update method (SR1)* belongs to the Broyden family, this by setting $\phi_k = \frac{s_k^T y_k}{s_k^T y_k - s_k^T B_k s_k}$. However, it does not belong to the restricted class, since ϕ_k may be outside the interval $[0, 1]$ [22, p.209]. The SR1 update matrix U_k takes the following form:

- Symmetric rank one method (SR1)

$$U_k = \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}, \tag{3.6}$$

see, e.g., [19, p.351] and [22, p.202]. It was originally discovered by Davidon [5], according to [21]. One fact of SR1 is that even if B_k is positive definite, the update B_{k+1} may not have this property. However, trust-region methods solve that. A real disadvantage is that there might be steps, when no update satisfy the secant condition (3.3). As long as the denominator is different from zero the method proceeds with a unique rank-one update. If $y_k = B_k s_k$ the only update that satisfies the secant condition is $U_k = 0$, such that the same B matrix can be used another iteration. The failure occurs when $y_k \neq B_k s_k$ and $(y_k - B_k s_k)^T s_k = 0$ at the same iteration. All desired characteristics can not be fulfilled and a rank two correction has to be used. Note that this failure can occur with an objective function that is convex and quadratic [22, pp.202-203]. One example on when the SR1 method fail is for a convex quadratic function dependent of only two variables, see Appendix B1 for derivation,

$$\begin{cases} f(x) = \frac{1}{2} x^T \begin{pmatrix} 2 & 0 \\ 0 & 1/2 \end{pmatrix} x, \\ \text{Initial x-value: } x_0 = \begin{pmatrix} \sqrt{2} \\ 8 \end{pmatrix}. \end{cases} \tag{3.7}$$

The SR1 method fails during the first update on the problem (3.7) in exact arithmetic. However, due to numerical instability an implemented SR1 method is able to solve the problem (3.7) without failing. In Appendix B2 a problem dependent of three variables is presented, which cause the SR1 method to fail both in exact arithmetic and as implemented method. A strategy to avoid break down is desirable. One would be to use the update (3.6) when the denominator is large enough and otherwise skip the update by letting $B_{k+1} = B_k$. A strategy would be to apply the update (3.6) only if $|s_k^T(y_k - B_k s_k)| \geq \delta ||s_k|| \, ||y_k - B_k s_k||$, where $\delta \in (0, 1)$ is a small number. If this criteria does not hold, let $B_{k+1} = B_k$. Implementations of the SR1 often use a skipping rule of this type [22, pp.203-204]. However, this skipping strategy might prevent the method from converging rapidly. One example is the problem given in Appendix B2, which perform as the steepest descent method since the Hessian approximation continue to equal the identity matrix during the iterations. Another strategy is to perform a rank two update whenever both $y_k \neq B_k s_k$ and $(y_k - B_k s_k)^T s_k = 0$ at the same iteration. Below two different rank two updates within the Broyden family are presented.

BFGS and *DFP* are two other update methods belonging to the Broyden family. First the BFGS method is presented, and then the DFP method is outlined. Finally relations between the two methods and the Broyden family are presented.

The BFGS method is named after its developers Broyden, Fletcher, Goldfarb and Shanno [19, p.355]. By setting $\phi_k = 0$ in (3.5), the BFGS formula is obtained. It is believed that the most effective update of the Broydan family is the BFGS method [11, p.119]. The BFGS update matrix U_k takes the following form:

- Broyden, Fletcher, Goldfarb and Shanno method (BFGS)

$$U_k = -\frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \tag{3.8}$$

see, e.g., [19, p.355]. For the BFGS update method with Wolfe line search (4.6) Powell [28] proves global convergence for a convex objective function [8, p.68]. Byrd, Nocedal and Yuan [3] extend Powell's result for all $\phi_k \in [0, 1)$.

A third update method of the Broyden family is the DFP method, which is named after its developers Davidon, Fletcher and Powell. The DFP update formula is obtained by setting $\phi_k = 1$. The update matrix U_k of the DFP method takes the following form:

- Davidon, Fletcher and Powell method (DFP)

$$U_k = -\frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + (s_k^T B_k s_k) v_k v_k^T, \tag{3.9}$$

where v_k is given in (3.5) [19, p.355].

Infact the BFGS and DFP methods both have symmetric rank two updates [18, p.269]. From the Broyden family update (3.5) BFGS is obtained by setting $\phi_k = 0$ and DFP by setting $\phi_k = 1$. The update matrix of the Broyden family can be rewritten as a linear combination of these two methods,

$$B_{k+1} = (1 - \phi_k) B_{k+1}^{BFGS} + \phi_k B_{k+1}^{DFP}.$$

The two methods, BFGS and DFP, preserve positive definiteness of the Hessian approximations when the curvature condition (3.4) is fulfilled. This relation implies that the restricted Broyden family, i.e. $0 \leq \phi \leq 1$, has the same property [22, p.208]. It can be observed that

$$U_k = U_k^{BFGS} + \phi_k (s_k^T B_k s_k) v_k v_k^T,$$

where v_k is given in (3.5). A rewritten formula of the Broyden family, where the parameter ϕ_k is an arbitrary real value. It should be noted that for a general problem the curvature condition (3.4) might not hold, and therefore the BFGS method can not always be used to perform the update. A strategy is to use the SR1 update when BFGS can not be applied [21]. However, there is no guarantee that this update is possible either.

3.3. Algorithm

Formulas have been presented above on how to calculate various important parameters within the quasi-Newton method, at a certain iteration k. To summarize the main parts of the method, one version is presented on the following page as an algorithm.

Algorithm 3.1. *The quasi-Newton algorithm*

$k \leftarrow 0$; $x_k \leftarrow$ initial point;
$B_k \leftarrow$ initial Hessian approximation;
while $\|\nabla f(x_k)\|_2 \neq 0$ **then**
  $p_k \leftarrow$ the solution of $B_k p_k = -\nabla f(x_k)$;
  $\alpha_k \leftarrow$ line search method: (4.1), (4.3), Algorithm 4.1, 4.2 or 4.3;
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
  $s_k \leftarrow x_{k+1} - x_k$;
  $y_k \leftarrow \nabla f(x_{k+1}) - \nabla f(x_k)$;
  $U_k \leftarrow$ update method: (3.6), (3.8) or (3.9);
  $B_{k+1} \leftarrow B_k + U_k$;
  $k \leftarrow k + 1$;
end

In this algorithm all variables are written with subscripts. It is only the current values of the variables that is needed to be saved, with two exceptions. The exceptions are the previous x-value and the previous gradient, since needed in the update of s and y.

3.4. Self-scaling quasi-Newton methods

The general strategy of self-scaling quasi-Newton method (SS) is to scale the Hessian approximation matrix B_k before it is updated at each iteration. This is to avoid large difference in the eigenvalues of the approximated Hessian of the objective function. Self-scaling variable metric algorithms was introduced by Oren, see [24], [25] and [26]. The Hessian approximation matrix B_k can be updated according to a self-scaling BFGS update of the form

$$B_{k+1} = \rho_k[B_k - \frac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k}] + \frac{y_k y_k^T}{y_k^T s_k}, \tag{3.10a}$$

$$\rho_k = \frac{y_k^T s_k}{s_k^T B_k s_k}, \tag{3.10b}$$

where ρ_k is the self-scaling factor. For a general convex objective function, Nocedal and Yuan proves global convergence of a SS-BFGS (3.10) with Wolfe line search (4.6) [23]. They also present results indicating that the unscaled BFGS method in general is superior to the SS-BFGS (3.10), with its ρ_k of Oren and Luenberger [26]. A suggestion of Al-Baali, see [1] and [2], is to modify the self-scaling factor to

$$\rho_k = \min\{\frac{y_k^T s_k}{s_k^T B_k s_k}, 1\}. \tag{3.11}$$

This modification of ρ_k gives a global convergent SS-BFGS method which is competitive with the unscaled BFGS method. If applying this self-scaling quasi-Newton

method on the Broyden family, it takes the form

$$\begin{cases} B_{k+1} = \rho_k(B_k - \frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k} + \phi_k(s_k^T B_k s_k) v_k v_k^T) + \frac{y_k y_k^T}{y_k^T s_k}, \\ v_k = \frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k}, \\ \rho_k = \min\{\frac{y_k^T s_k}{s_k^T B_k s_k}, 1\}. \end{cases}$$

This method reduces to the unscaled Broyden family if $\rho_k = 1$, or equivalently replacing the scaled matrix $\rho_k B_k$ to the unscaled matrix B_k. To implement the self-scaling quasi-Newton in an algorithm, only a smaller modification has to be made in the Algorithm 3.1. After updating the y_k add the update $B_k \leftarrow \rho_k B_k$ before updating the U_k. This modification turn the algorithm into a self-scaling quasi-Newton algorithm, where ρ_k is given by (3.10b) or (3.11).

3.5. Adapt for large problems

One strategy for solving large problems is to let the last few iterations define a variable metric approximation of the Hessian, known as the limited-memory BFGS method. It store a certain number of pairs $\{s_i, y_i\}$ instead of storing the large Hessian approximation B_k. To obtain the search direction p_k a sequence of inner products are performed involving $\nabla f(x_k)$ and $\{s_i, y_i\}$. The oldest set within $\{s_i, y_i\}$ are replaced with the new calculated iterate. An improvement to the procedure would be to design a strategy for selecting the most useful correction pairs and not simply the most recent ones [21]. There also exist limited-memory reduced-Hessian methods, in which the dimension of the Hessian is reduced to save storage. Let $B_{k+1} = (p_m \dots p_k \ g_{k+1})$. The oldest point $m - 1$ is discarded before starting iteration $k + 1$ [10].

4. Line search methods

In this section of line search methods, different choices of stepsize α_k are presented and discussed. To perform a line search is to find a α_k that reduce the objective function f, but not spend too much time tracking it. Ideal is to find the stepsize α_k which perform a line search that minimizes the objective function, known as exact line search [22, p.36].

4.1. Exact line search

To perform exact line search on a problem is to take the best possible step α_k for a given search direction p_k. For a general problem it is not known how to analytically perform exact line search. However, a well known problem that exact line search can be applied on is the quadratic problem (1.2). To perform exact line search on this problem is to let the stepsize

$$\alpha_k = -\frac{\nabla f(x_k)^T p_k}{p_k^T H p_k}, \tag{4.1}$$

for a given point x_k and search direction p_k [11, p.145].

Another problem that it is possible to perform exact line search on is

$$f(x) = \gamma(x^T x)^2 + \frac{1}{2}x^T Hx + c^T x. \tag{4.2}$$

This problem is similar to the quadratic problem (1.2), with the only difference that a constant γ times a fourth degree term in x is added. This term could for example module disturbance on a quadratic problem. To be able to perform an exact line search the α_k that minimizes the stepsize of $f(x_k + \alpha_k p_k)$ has to be obtained. The calculations are presented in Appendix C. It turns out that the α_k is obtained by solving

$$\begin{cases} a_1\alpha_k^3 + a_2\alpha_k^2 + a_3\alpha_k + a_4 = 0, \text{ where} \\ a_1 = 4\gamma(p_k^T p_k)^2, \\ a_2 = 12\gamma(p_k^T p_k)(x_k^T p_k), \\ a_3 = 4\gamma((x_k^T x_k)(p_k^T p_k) + 2(x_k^T p_k)^2) + p_k^T H p_k, \\ a_4 = 4\gamma(x_k^T x_k)(x_k^T p_k) + (x_k^T H p_k + c^T p_k). \end{cases} \tag{4.3}$$

Let α_k be the real α_k-value obtained by solving the equation (4.3). Hence the function (4.2) is convex, there will only exist one real solution. Note that within this research equation (4.3) was solved by the function *roots* in Matlab.

4.2. Backtracking line search

Backtracking line search is a strategy which starts from an initial stepsize guess and successively decrease the size of α_k until a function reduction is obtained [21]. The strategy is presented in the algorithm below.

Algorithm 4.1. *Backtracking line search algorithm*

$\alpha_k \leftarrow 1$; $iter \leftarrow 0$; $itermax \leftarrow$ maximal number of iterations;
while $iter < itermax$ **then**
  $iter \leftarrow iter + 1$;
  $\tilde{x} \leftarrow x_k + \alpha_k p_k$;
  if $f(\tilde{x}) < f(x_k)$ **then**
    Stop;
  else
    $\alpha_k \leftarrow \alpha_k/2$;
  end
end

Note that 20 was the maximal number of iterations within this research. A modified backtracking line search for the objective function (4.2) is to let the initial stepsize α_k gets (4.1).

4.3. Armijo's rule

To terminate a line search the criteria of Armijo's rule can be applied. The strategy is to first guarantee that the selected stepsize α_k is not too large, and then that it is not be to small. Define the function

$$\phi(\alpha_k) = f(x_k + \alpha_k p_k). \tag{4.4}$$

A stepsize α_k is considered to be not too large if

$$\phi(\alpha_k) \leq \phi(0) + \sigma_1 \phi'(0)\alpha_k, \tag{4.5}$$

and is considered to be not to small if

$$\phi(\alpha_k \eta) > \phi(0) + \sigma_1 \phi'(0)\alpha_k \eta.$$

The constants has to fulfill following criteria $0 < \sigma_1 < 1$ and $\eta > 1$. To begin the search an arbitrary α_k can be used, while σ_1 and η are chosen within the allowed range, e.g. let $\alpha_k \leftarrow 1$, $\sigma_1 \leftarrow 0.2$ and $\eta \leftarrow 2$. The Armijo line search can be performed according to the Algorithm 4.2 below [18, p.212].

Algorithm 4.2. *Armijo's line search algorithm*

$\alpha_k \leftarrow 1; \quad \sigma_1 \leftarrow 0.2; \quad \eta \leftarrow 2;$
if $\phi(\alpha_k) \leq \phi(0) + \sigma_1 \phi'(0)\alpha_k$ **then**
  while $\phi(\alpha_k) \leq \phi(0) + \sigma_1 \phi'(0)\alpha_k$ **then**
    $\alpha_k \leftarrow \eta \alpha_k;$
  end
  $\alpha_k \leftarrow \alpha_k / \eta;$
else
  while $\phi(\alpha_k) > \phi(0) + \sigma_1 \phi'(0)\alpha_k$ **then**
    $\alpha_k \leftarrow \alpha_k / \eta;$
  end
end

4.4. Wolfe conditions

Another line search method is to find a stepsize α_k which fulfill the Wolfe conditions. The stepsize α_k is accepted if it satisfies the two Wolfe conditions

$$\phi(\alpha_k) \leq \phi(0) + \sigma_1 \alpha_k \phi'(0), \tag{4.6a}$$
$$\phi'(\alpha_k) \geq \sigma_2 \phi'(0), \tag{4.6b}$$

where the function $\phi(\alpha_k)$ is defined according to (4.4). The two constants σ_1 and σ_2 ought to fulfill $0 < \sigma_1 < \sigma_2 < 1$. These two relations (4.6) are known as the Wolfe conditions [21]. Let us include the criterium $0 < \sigma_1 < \frac{1}{2}$ to the Wolfe conditions (4.6). Note that the first Wolfe condition (4.6a) is the Armijo condition (4.5), while the second Wolfe condition (4.6b) is a curvature condition. The curvature

condition can be modified to force the stepsize α_k into a broad neighborhood of a local minimizer or stationary point of $\phi(\alpha_k)$ [22, p.39]. These modifications are known as the strong Wolfe conditions

$$\phi(\alpha_k) \leq \phi(0) + \sigma_1 \alpha_k \phi'(0), \tag{4.7a}$$
$$-|\phi'(\alpha_k)| \geq \sigma_2 \phi'(0), \tag{4.7b}$$

where $0 < \sigma_1 < \sigma_2 < \frac{1}{2}$. Note that a stepsize α_k satisfying the strong Wolfe conditions (4.7) also satisfies the usual Wolfe conditions (4.6). A line search method which for a convex function will find a α_k that fulfills the Wolfe conditions is to be presented. Define

$$\begin{cases} g(\alpha_k) = \phi(\alpha_k) - \phi(0) - \sigma_2 \alpha_k \phi'(0), \\ g'(\alpha_k) = \phi'(\alpha_k) - \sigma_2 \phi'(0). \end{cases}$$

For a convex function at a point x_k, along a descent direction p_k, the function value $\phi'(0) < 0$, and therefore $g'(0) < 0$. Though it is not possible that $g'(\alpha_k) < 0$ $\forall \alpha_k \geq 0$, since the function is convex. Hence, it is not possible that $\phi'(\alpha_k) < \sigma_2 \phi'(0)$ $\forall \alpha_k \geq 0$. This implies, since the function is continuous, that there will be a stepsize $\tilde{\alpha}_k$ where

$$g'(\tilde{\alpha}_k) = 0, \text{ i.e. } \phi'(\tilde{\alpha}_k) = \sigma_2 \phi'(0).$$

This stepsize $\tilde{\alpha}_k$ fulfills (4.6b). This formulation also fulfills (4.6a) because

$$\phi(\tilde{\alpha}_k) - \phi(0) - \sigma_1 \tilde{\alpha}_k \phi'(0) < \phi(\tilde{\alpha}_k) - \phi(0) - \sigma_2 \tilde{\alpha}_k \phi'(0) = g(\tilde{\alpha}_k) < g(0) = 0.$$

This formulation fulfills the Wolfe conditions (4.6). Note that the parameter σ_2 has an essential role in how close $\phi'(\tilde{\alpha}_k)$ is to zero. If finding a point b where $g'(b) > 0$, it would be possible to perform an interval reduction between [a,b] to locate $\tilde{\alpha}_k$, since $a = 0$ fulfills $g'(a) < 0$. An algorithm that as a first step finds a b which fulfills $g'(b) > 0$ and as second step perform interval reduction on [a,b] is presented below.

Algorithm 4.3. *Wolfe line search algorithm, Step 1 and Step 2*

$a \leftarrow 0$; $\sigma_1 \leftarrow 10^{-3}$; $\sigma_2 \leftarrow 10^{-2}$ $\epsilon \leftarrow 10^{-9}$;
Step 1: Find a valid b.
$\alpha_k \leftarrow -g'(0)/g''(0)$; $\Delta_1 \leftarrow 0$; $\Delta_2 \leftarrow 0$;
while $\Delta_1 = 0$ **then**
  if $g'(\alpha_k) < -\epsilon$ **then**
    $a \leftarrow \alpha_k$;
    $\alpha_k \leftarrow 2\alpha_k$;
  else if $g'(\alpha_k) > \epsilon$ **then**
    $b \leftarrow \alpha_k$; $\Delta_1 \leftarrow 1$;
  else
    $\Delta_1 \leftarrow 1$; $\Delta_2 \leftarrow 1$;
  end
end

Step 2: Find $\tilde{\alpha}_k$ by interval reduction.
while $\Delta_2 = 0$ **then**
  if $g'(\alpha_k) < -\epsilon$ **then**
    $a \leftarrow \alpha_k$;
    $\alpha_k \leftarrow a + \frac{1}{2}(b-a)$;
  else if $g'(\alpha_k) > \epsilon$ **then**
    $b \leftarrow \alpha_k$;
    $\alpha_k \leftarrow a + \frac{1}{2}(b-a)$;
  else
    $\Delta_2 \leftarrow 1$;
  end
end
$\tilde{\alpha}_k \leftarrow \alpha_k$

There also exist line search algorithms which track a stepsize α_k fulfilling the strong Wolfe conditions (4.7), e.g. in Nocedal [22, pp.58-60].

5. Computing environment

The computing environment is a computer using the software Matlab 7.0.1.15 (R14) Service Pack 1 September 13, 2004. The investigations of the methods are exclusively performed in Matlab, by applying algorithms on testproblems.

6. Numerical results

The results of the research are to be presented in this section. The problems to be solved are unconstrained optimization formulations of the form (1.1). Below, a general algorithm used within this study is presented.

Algorithm 6.1. *General algorithm*

$k \leftarrow 0$; $x_k \leftarrow$ initial point;
$sc \leftarrow$ stop criteria value; $itermax \leftarrow 999$;
while $\|\nabla f(x_k)\|_2 \geq sc$ **and** $k < itermax$ **then**
  $p_k \leftarrow$ CG section 2 or QN section 3;
  $\alpha_k \leftarrow$ line search method section 4;
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
  $k \leftarrow k + 1$;
end

The stop criteria is when $||\nabla f(x_k)||_2 = \sqrt{\nabla f(x_k)^T \nabla f(x_k)} < sc$. The sc is a specified convergence tolerance. In this research the stopping criteria had following values: $sc = [1, 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}]$. Note that the initial Hessian approximation matrix in quasi-Newton equals the identity matrix, $B_0 = I$. The testproblems and line search methods considered within this research are listed on the following page.

Testproblems:

1. The convex quadratic objective function from (1.2),

$$f(x) = \frac{1}{2}x^T Hx + c^T x.$$

2. The convex perturbed quadratic objective function from (4.2),

$$f(x) = \gamma(x^T x)^2 + \frac{1}{2}x^T Hx + c^T x.$$

3. The convex random Q perturbed quadratic objective function

$$f(x) = (x^T Qx)^2 + \frac{1}{2}x^T Hx + c^T x,$$

where Q is a symmetric positive definite random matrix. This random matrix is created according to an algorithm presented in Appendix D.

All testproblems can be modified by vary H, c and the initial x-value x_0. The different variations investigated within this report are

a) $H = diag(m,\, m-1,\, \ldots,\, 1),\ c = (1, \ldots,\, 1)^T,\ x_0 = (0, \ldots,\, 0)^T,$

b) $H = diag(10m,\, 5m,\, m,\, m-1,\, \ldots,\, 1),\ c = (1, \ldots,\, 1)^T,\ x_0 = (0, \ldots,\, 0)^T,$

for $m = 10$, 100 and 1000.

Line search methods:

- Exact line search. Testproblem 1 has (4.1) and Testproblem 2 has (4.3).
- Backtracking line search. Algorithm 4.1 for all testproblems.
- Wolfe line search. Algorithm 4.3 for all testproblems.

In the graphs to be presented the x-axis is numbered from 1 to 6. Each number represents a method, where 1-3 represents CG methods and 4-6 represents QN methods. The methods are: 1) FR; 2) HS; 3) PR; 4) BFGS; 5) DFP; 6) SR1. The y-axis represents the number of iterations performed on a given problem. Each stop criteria sc is represented by its own color in the graphs, switch of color indicate that a harder criteria is aimed to be fulfilled.

6.1. Testproblem 1

The study has originated from Testproblem 1 where $f(x)$ is a quadratic function. The first formulation to be investigated is a), which is denoted by Testproblem 1a. We perform exact line search, backtracking line search and Wolfe line search on Testproblem 1a for $m = 10$, 100 and 1000.

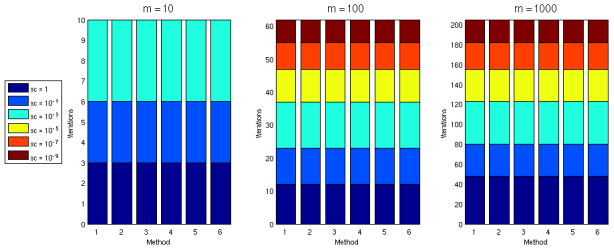

Figure 1: Testproblem 1a: Exact line search

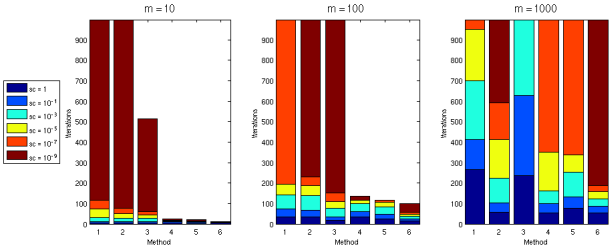

Figure 2: Testproblem 1a: Backtracking line search

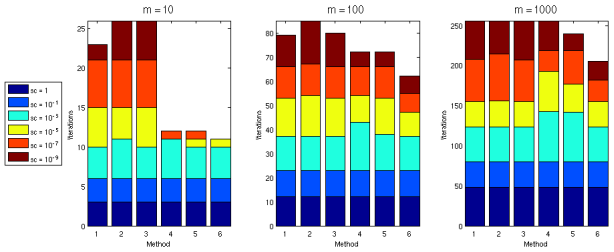

Figure 3: Testproblem 1a: Wolfe line search

The second formulation to be investigated is b), which is denoted by Testproblem 1b. We perform exact line search, backtracking line search and Wolfe line search on Testproblem 1b for $m = 10$, 100 and 1000.

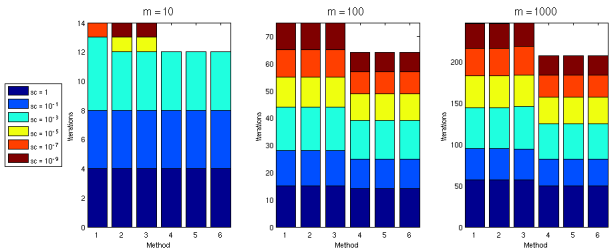

Figure 4: Testproblem 1b: Exact line search

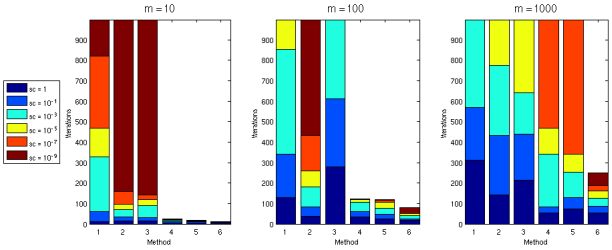

Figure 5: Testproblem 1b: Backtracking line search

The SR1 update method failed to find the optimum of Testproblem 1b, for $m = 1000$, while using backtracking line search.

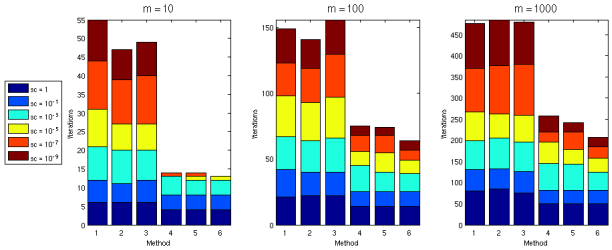

Figure 6: Testproblem 1b: Wolfe line search

6.2. Testproblem 2

Testproblem 2 is of particular interest since an exact line search has been derived for this problem, see Apendix B. The testproblem is investigated with different values of the constant $\gamma = [0,\ \lambda_{max}/2,\ \lambda_{max}]$, where the largest eigenvalue of the Hessian H is denoted as λ_{max}. The first formulation to be investigated is Testproblem 2a. We perform exact line search, backtracking line search and Wolfe line search on Testproblem 2a for $m = 10$, 100 and 1000. The results for $\gamma = 0$ is not presented. This is due to the fact that these results are very similar to the results of Testproblem 1a.

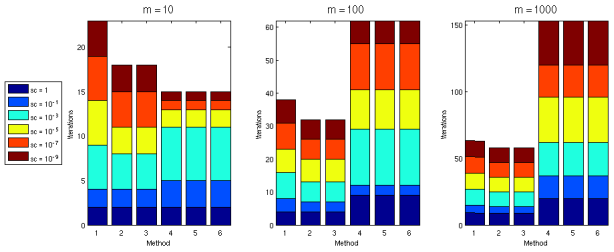

Figure 7: Testproblem 2a: Exact line search, with $\gamma = \lambda_{max}/2$

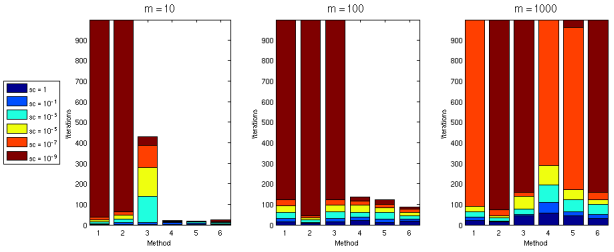

Figure 8: Testproblem 2a: Backtracking line search, with $\gamma = \lambda_{max}/2$

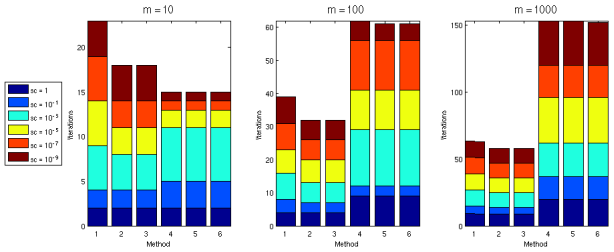

Figure 9: Testproblem 2a: Wolfe line search, with $\gamma = \lambda_{max}/2$

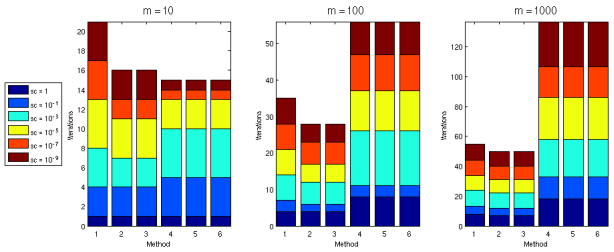

Figure 10: Testproblem 2a: Exact line search, with $\gamma = \lambda_{max}$

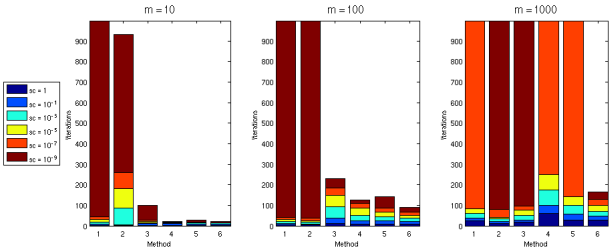

Figure 11: Testproblem 2a: Backtracking line search, with $\gamma = \lambda_{max}$

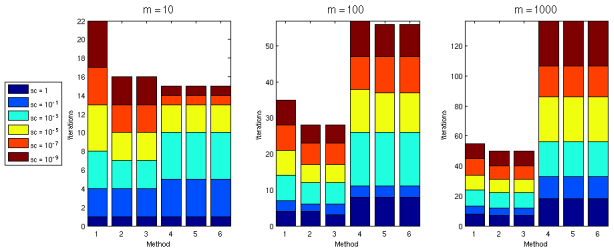

Figure 12: Testproblem 2a: Wolfe line search, with $\gamma = \lambda_{max}$

The second formulation to be investigated is b), which is denoted by Testproblem 2b. We perform exact line search, backtracking line search and Wolfe line search on Testproblem 2b for $m = 10$, 100 and 1000. These tests are performed for each of the three γ-values. However, the results for $\gamma = 0$ is not presented. This is due to the fact that these results are very similar to the results of Testproblem 1b.

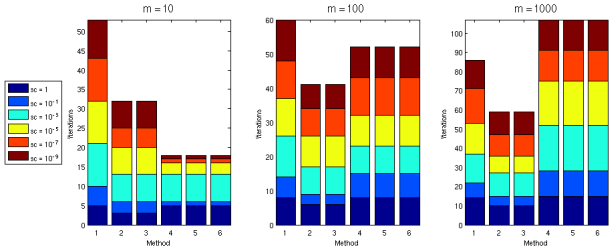

Figure 13: Testproblem 2b: Exact line search, with $\gamma = \lambda_{max}/2$

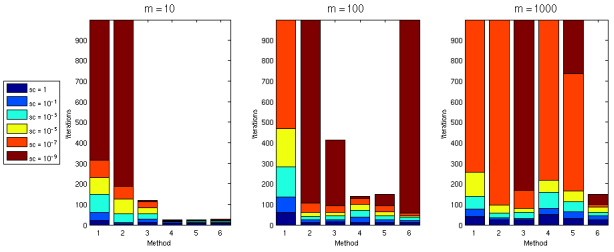

Figure 14: Testproblem 2b: Backtracking line search, with $\gamma = \lambda_{max}/2$

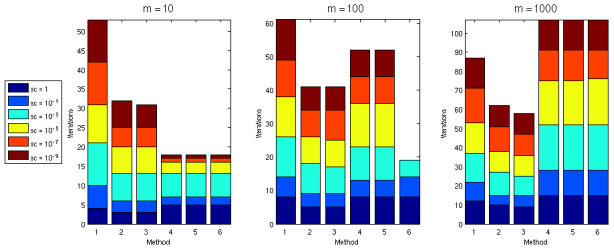

Figure 15: Testproblem 2b: Wolfe line search, with $\gamma = \lambda_{max}/2$

The SR1 update method failed to find the optimum of Testproblem 2b with $\gamma = \lambda_{max}/2$, for $m = 100$, while using Wolfe line search.

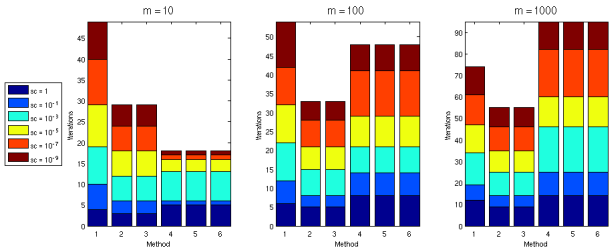

Figure 16: Testproblem 2b: Exact line search, with $\gamma = \lambda_{max}$

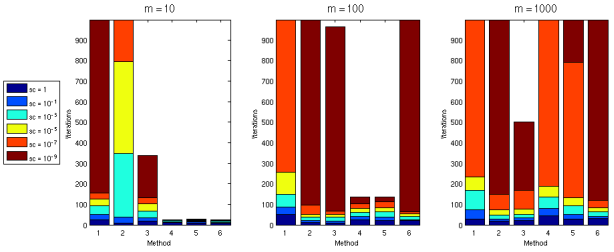

Figure 17: Testproblem 2b: Backtracking line search, with $\gamma = \lambda_{max}$

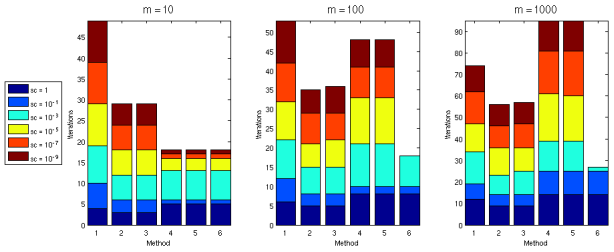

Figure 18: Testproblem 2b: Wolfe line search, with $\gamma = \lambda_{max}$

The SR1 update method failed to find the optimum of Testproblem 2b with $\gamma = \lambda_{max}$, for $m = 100$ and for $m = 1000$, while using Wolfe line search.

6.3. Testproblem 3

Testproblem 3 is also a modification of the quadratic Testproblem 1. The difference is a added squared quadratic term $(x^T Qx)^2$, where the matrix Q is a convex random matrix. The first formulation to be investigated is Testproblem 3a. We perform backtracking line search, Wolfe line search and tightened Wolfe line search for Testproblem 3a when $m = 10$, 100 and 1000. Tightened Wolfe line search is Algorithm (4.3) with the modification that $\sigma_1 \leftarrow 10^{-5}$ and that $\sigma_2 \leftarrow 10^{-4}$.

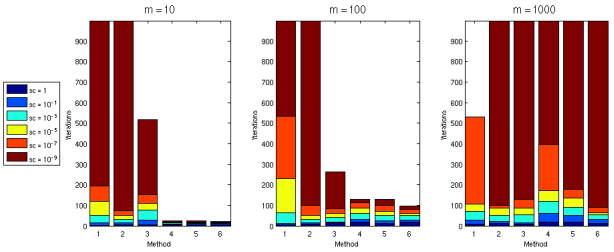

Figure 19: Testproblem 3a: Backtracking line search

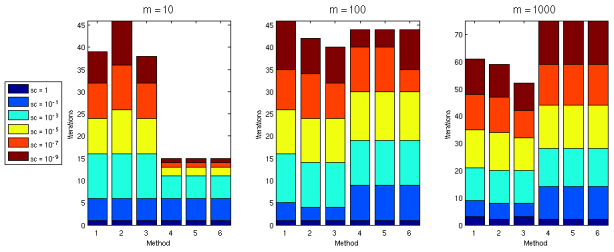

Figure 20: Testproblem 3a: Wolfe line search

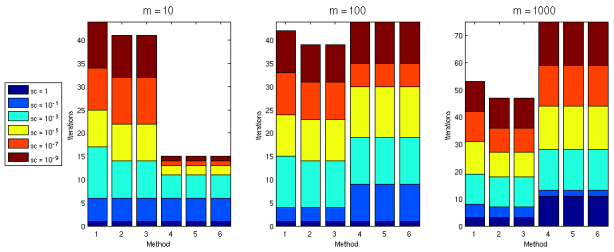

Figure 21: Testproblem 3a: tightened Wolfe line search

The second formulation to be investigated is Testproblem 3b. We perform backtracking line search, Wolfe line search and tightened Wolfe line search for Testproblem 3b when $m = 10$, 100 and 1000.

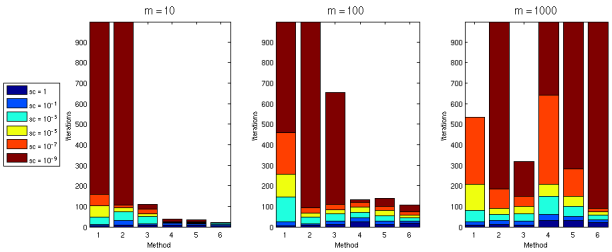

Figure 22: Testproblem 3b: Backtracking line search

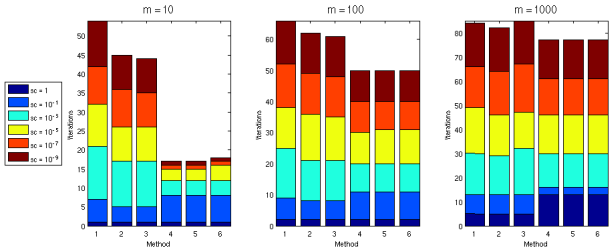

Figure 23: Testproblem 3b: Wolfe line search

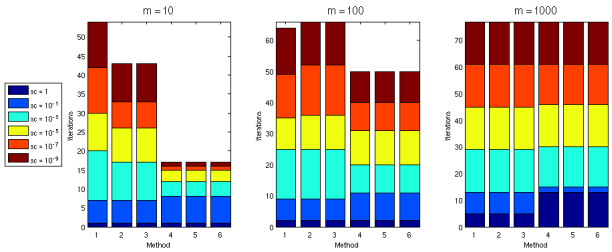

Figure 24: Testproblem 3b: tightened Wolfe line search

7. Discussion

On Testproblem 1 the QN method is preferable. The three methods of the quasi-Newton methods perform better than the conjugate-gradient methods for the different line search methods used on this problem. The exception is when the SR1 method fail on Testproblem 1b with backtracking line search and $m = 1000$. With exact line search applied on Testproblem 1 the methods are mathematically equivalent, which can be noted for formulation a). Though, for formulation b) numerical rounding errors cause the conjugate-gradient methods to perform worse. When the problem is quadratic and exact line search is applied this implicate that the generated vectors of the CG are orthogonal in theory. In Testproblem 1b, this exact orthogonality is only observed at the beginning of the process. At some stage the generated vectors starts losing their global orthogonality due to rounding errors. The CG and QN diverge from each other and eventually different number of iterations can be observed, even though equivalent in theory.

For Testproblem 2 it is the reversed situation. For this problem the CG methods seem to be preferable, especially the Polak-Ribière method. CG is preferable when applying Testproblem 2 to exact line search or Wolfe line search, especially for larger m. If applying backtracking line search, some of the QN methods or the PR method is to prefer. Note that the SR1 method failed on Testproblem 2b with $m = 100$ both for $\gamma = \lambda_{max}/2$ and $\gamma = \lambda_{max}$, when Wolfe line search was applied. With these two SR1 failings as exceptions, the difference between the exact line search and the Wolfe line search is small.

On Testproblem 3 the CG tend to be slightly more preferable on a) while QN tend to be slightly more preferable on b).

The results suggest that if using the conjugate-gradient method, then the update method should be the Polak-Ribière method or Hestenes-Stiefel method. These two methods often perform very similarly when the applied line search method is Wolfe or exact. If applying the backtracking line search the performance differ in a greater extent, with a slight advantage for the Polak-Ribière method. The Fletcher-Reeves method often need some extra iterations compared to the other conjugate-gradient methods. The results on the behavior of the conjugate-gradient methods seem to agree with previous known results.

Note that the quasi-Newton methods might have some difficulty with Testproblem 2 and Testproblem 3 because these problems have a fourth degree term. This term might disturb the forming of the Hessian approximation while solving the problem. The results indicate that the two most preferable quasi-Newton methods are the BFGS method and the DFP method. The BFGS method and the DFP method perform very similarly. A well known theoretical aspect favor the BFGS method. For a convex objective function the BFGS has global convergence with inexact searches subject to some conditions. This knowledge is of great interest while solving a problem. The BFGS method and the DFP method seem to be more stable than the SR1 method. The symmetric rank one method sometimes perform very well, but on the other hand it sometimes breakdown and is unable to locate the optimum. The breakdowns occur when the SR1 update matrix turns indefinite.

It is then possible to shifting up the eigenvalues of the update matrix, by adding a scalar times the identity matrix, to make the update matrix positive definite again. If this procedure is performed each time the update matrix turns indefinite the SR1 method is able to solve the problems presented as breakdowns within this research. Another known strategy is to apply trust-region methods, which prevent the SR1 update matrix to become indefinite. The results on the behavior of the quasi-Newton methods seem to agree with previous known results, though this research is unable to favor the BFGS method compared to the DFP method by the results of the performed research. If the problem to be solved has large difference in the eigenvalues of the Hessian approximation, a self-scaling quasi-Newton method might decrease the number of iterations significantly compared to a quasi-Newton method. A well known theoretical result favor the SS-BFGS method. For a general convex objective function with inexact line search subject to some conditions the SS-BFGS method has global convergence. For large problems it is of high interest to decrease the number of stored elements of the Hessian approximation. A further study on the limited memory quasi-Newton methods is then recommended.

The results also suggest that unless an exact line search is available, the Wolfe line search seem to be preferable compared to the backtracking line search. The choice of line search method might affect the total number of iterations in a very great extent.

It is not recommended to draw too strong conclusions from the numerical results of this research. This is due to the fact that the performed investigation is a rather small study. Another important fact is that the computational time has been neglected, which might be a crucial factor for some applications.

8. Conclusion

It can be concluded that none of the methods should be totally excluded. The best method seems to depend on which problem that is to be solved. Within the conjugate-gradient methods, the Polak-Ribière method or alternatively the Hestenes-Stiefel method tend to be preferable according to the results. The situation between the quasi-Newton methods is more unclear. The BFGS method and the DFP method perform similarly, while the SR1 method sometimes perform very well and sometimes breakdown, though the breakdowns can be prevented by e.g. trust-region methods. For a convex objective function the BFGS method with an inexact search subject to some conditions guarantees global convergence according to theory, which favor the BFGS method.

Which method is then to favor in an application? If a problem with convex objective function is to be solved, it is probably advisable to use the mentioned BFGS method with the line search method that guarantees global convergence. If a problem is going to be solved repeatedly within some application, it might be of interest to perform a similar comparison of the methods. If the solution time is crucial the fastest method obviously should be used. If the accuracy of the result is of great importance, the problem can be solved by more than one method, e.g. solve with both the PR method and the BFGS method.

Appendix A: Solving a linear system of equations

Alternatives on how to solve a linear system of equations $Ax = b$, where A is a symmetric and positive definite matrix of size $n \times n$ is to be presented.

A1: Cholesky factorization

A symmetric positive definite matrix A has a unique lower triangular matrix C with positive diagonal elements such that $A = CC^T$, known as the Cholesky factorization of A. To solve a linear system of equations $Ax = b$ by the Cholesky factorization can be made in two steps. The first step is to solve $Cy = b$ for y by forward substitution. The second step is to solve $C^T x = y$ for x by backward substitution. An algorithm to calculate the Cholesky factorization matrix C is presented below [15, pp.204-205].

Algorithm 8.1. *Cholesky factorization of A*

for $i = 1, \ldots, n$ **do**
  for $j = 1, \ldots, i-1$ **do**
    $c_{i,j} \leftarrow (a_{i,j} - \sum_{k=1}^{j-1} c_{i,k}c_{j,k})/c_{j,j}$;
  end
  $c_{i,i} \leftarrow \sqrt{a_{i,i} - \sum_{k=1}^{i-1} c_{i,k}^2}$;
end

A2: Householder method

A Householder rotation of A transforms the matrix into the form QR, where R is a $m \times m$ upper triangular matrix and $Q^T Q = I$ [29, pp.13-14]. To solve a linear system of equations $Ax = b$ by the Householder rotation can be done in two steps. The first step is to multiply with Q^T on $QRx = b$. The second step is to solve $Rx = Q^T b$ for x by backward substitution. A strategy on how to find the Householder rotation is presented below. An alternative to orthogonalizing a sequence of vectors is by using Householder reflectors P_k of the form

$$P_k = I - 2w_k w_k^T,$$

where the vector w is of 2-norm unity. The vector is given by

$$\begin{aligned} w_k &= \tfrac{z}{||z||_2}, \\ z_i &= \begin{cases} 0 & \text{if } i < k, \\ \beta + a_{ii} & \text{if } i = k, \\ a_{ik} & \text{if } i > k, \end{cases} \\ \beta &= sign(a_{kk})(\textstyle\sum_{i=k}^{n} a_{ik}^2)^{1/2}. \end{aligned}$$

The $m \times n$ matrix A, with $m \leq n$, has its first column transformed into a multiple of e_1 by multiply $P_1 A$. By applying $m - 1$ Householder transforms onto A, it is reduced to upper triangular form

$$P_{m-1} P_{m-2} \ldots P_1 A = \begin{pmatrix} R \\ O \end{pmatrix},$$

where R is a $m \times m$ upper triangular matrix and O is a $(n-m) \times m$ zero block. Let $Q = P^T E_m$, where E_m is the matrix that consists of the first m columns of the identity matrix I. The sought matrices R and Q fulfill $A = QR$ [29, pp.11-14].

A3: Lanczos method

A method for solving the linear system of equations $Ax = b$ is the Lanczos algorithm which first appeared in 1950 [17]. One version of the Lanczos algorithm is presented below [12, p.345].

Algorithm 8.2. *The Lanczos algorithm*

$\beta_0 \leftarrow \|b\|_2; \quad q_0 \leftarrow 0; \quad q_1 \leftarrow b/\beta_0; \quad \alpha_1 \leftarrow q_1^T A q_1;$
$d_1 \leftarrow \alpha_1; \quad c_1 \leftarrow q_1; \quad x_1 \leftarrow b/\alpha_1;$
for $j = 1, \ldots, n-1$ **do**
  $r_j \leftarrow (A - \alpha_j I) q_j - \beta_{j-1} q_{j-1};$
  $\beta_j \leftarrow \|r_j\|_2;$
  if $\beta_j = 0$ **then**
    $x \leftarrow x_j;$
    Stop;
  else
    $q_{j+1} \leftarrow r_j/\beta_j; \quad \alpha_{j+1} \leftarrow q_{j+1}^T A q_{j+1};$
    $\mu_{j+1} \leftarrow \beta_j/d_j; \quad d_{j+1} \leftarrow \alpha_{j+1} - \mu_{j+1}\beta_j;$
    $\rho_{j+1} \leftarrow -\mu_{j+1} d_j \rho_j / d_{j+1}; \quad c_{j+1} \leftarrow q_{j+1} - \mu_{j+1} c_j;$
    $x_{j+1} \leftarrow x_j + \rho_{j+1} c_{j+1};$
  end
end
$x \leftarrow x_n;$

Let $\rho_1 \leftarrow \frac{q_1^T b}{d_1}$ and $x_0 \leftarrow 0$ in the Lanczos method to get the equivalence for the initial choice of $x_0 \leftarrow 0$ in the conjugate-gradient method.

Appendix B: Failure of the symmetric rank one update method

Criteria of failure for the symmetric rank one update method, for a quadratic function dependent of m+1 variables when exact line search is applied is to be derived. Formulation of the problem,

$$f(x) = x(0)^2 + a_1 x(1)^2 + \ldots + a_m x(m)^2, \quad \text{Initial x-value: } x_0 = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_m \end{pmatrix},$$

where the vector $x = (\, x(0),\, x(1),\, \ldots,\, x(m)\,)^T$.

$$\nabla f(x) = 2 \begin{pmatrix} x(0) \\ a_1 x(1) \\ \vdots \\ a_m x(m) \end{pmatrix} \qquad \nabla^2 f(x) = 2 \begin{pmatrix} 1 & 0 & \ldots & \ldots \\ 0 & a_1 & 0 & \ldots \\ \vdots & 0 & \ddots & 0 \\ \vdots & \vdots & 0 & a_m \end{pmatrix} \succeq 0 \Rightarrow a_1 \geq 0, \ldots, a_m \geq 0$$

$$p_0 = -g_0 = -\nabla f(x_0) = -2 \begin{pmatrix} \beta_0 \\ a_1\beta_1 \\ \vdots \\ a_m\beta_m \end{pmatrix}$$

Exact line search,

$$\alpha = -\frac{p_0^T g_0}{g_0^T \nabla^2 f(x_0) g_0} = \frac{\beta_0^2 + a_1^2\beta_1^2 + \ldots + a_m^2\beta_m^2}{2(\beta_0^2 + a_1^3\beta_1^2 + \ldots + a_m^3\beta_m^2)} > 0$$

$$x_1 = \begin{pmatrix} \beta_0(1-2\alpha) \\ \beta_1(1-2\alpha a_1) \\ \vdots \\ \beta_m(1-2\alpha a_m) \end{pmatrix}, \quad g_1 = 2 \begin{pmatrix} \beta_0(1-2\alpha) \\ a_1\beta_1(1-2\alpha a_1) \\ \vdots \\ a_m\beta_1(1-2\alpha a_m) \end{pmatrix}$$

$$s = -2\alpha \begin{pmatrix} \beta_0 \\ a_1\beta_1 \\ \vdots \\ a_m\beta_m \end{pmatrix}, \quad y = -4\alpha \begin{pmatrix} \beta_0 \\ a_1^2\beta_1 \\ \vdots \\ a_m^2\beta_m \end{pmatrix}$$

The failure occur when $y_k \neq B_k s_k$ and $(y_k - B_k s_k)^T s_k = 0$ at the same iteration k. If $B_k = I$ and $y_k \neq s_k$ is fulfilled, then the other criteria $(y_k - B_k s_k)^T s_k = 0$ gives a condition when failure occur.

$$(y_k - s_k)^T s_k = 4\alpha^2(\beta_0^2 + a_1^2\beta_1^2(2a_1 - 1) + \ldots + a_m^2\beta_m^2(2a_m - 1)) = 0,$$

which is equivalent with

$$\beta_0^2 + a_1^2\beta_1^2(2a_1 - 1) + \ldots + a_m^2\beta_m^2(2a_m - 1) = 0.$$

When this condition and its premises are fulfilled, the SR1 update method is unable to find a symmetric rank one update for the stated problem in exact arithmetic.

Appendix B1: Failure of the SR1 update method in two dimensions

Criteria of failure for the symmetric rank one update method, for a quadratic function dependent of two variables when exact line search is applied is to be presented. Formulation of the problem,

$$f(x) = \frac{1}{2} x^T \begin{pmatrix} 2 & 0 \\ 0 & 2a_1 \end{pmatrix} x, \quad \text{Initial } x\text{-value: } x_0 = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}.$$

The condition is

$$\beta_0^2 + a_1^2\beta_1^2(2a_1 - 1) = 0.$$

For example chose: $a_1 = \frac{1}{4}$.
This choice gives $32\beta_0^2 = \beta_1^2$, where $\beta_0 = \sqrt{2}$ and $\beta_1 = 8$ is one possible choice.

Appendix B2: Failure of the SR1 update method in three dimensions

Criteria of failure for the symmetric rank one update method, for a quadratic function dependent of three variables when exact line search is applied is to be presented. Formulation of the problem,

$$f(x) = \frac{1}{2}x^T \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2a_1 & 0 \\ 0 & 0 & 2a_2 \end{pmatrix} x, \quad \text{Initial x-value: } x_0 = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix}.$$

The condition is

$$\beta_0^2 + a_1^2\beta_1^2(2a_1 - 1) + a_2^2\beta_2^2(2a_2 - 1) = 0.$$

For example chose: $a_1 = \frac{3}{4}$, $a_2 = \frac{1}{4}$ and $\beta_0 = 0$.
This choice gives $9\beta_1^2 = \beta_2^2$, where $\beta_1 = 1$ and $\beta_2 = 3$ is one possible choice.

Appendix C: Exact line search for perturbed quadratic problem

The exact line search α_k for $f(x) = \gamma(x^Tx)^2 + \frac{1}{2}x^THx + c^Tx$ is to be derived. For a given point x_k and search direction p_k the exact line search is found by solving

$$\underset{\alpha_k \in I\!R^1}{\text{minimize}} \quad f(x_k + \alpha_k p_k),$$

where the convex objective function is

$$\begin{aligned} f(x_k + \alpha_k p_k) = \quad & \gamma(p_k^Tp_k)^2\alpha_k^4 + 4\gamma(p_k^Tp_k)(x_k^Tp_k)\alpha_k^3 + [2\gamma((x_k^Tx_k)(p_k^Tp_k) + 2(x_k^Tp_k)^2) + \tfrac{1}{2}p_k^THp_k]\alpha_k^2 + ... \\ & ... + [4\gamma(x_k^Tx_k)(x_k^Tp_k) + (x_k^THp_k + c^Tp_k)]\alpha_k + [(x_k^Tx_k)^2 + \tfrac{1}{2}x_k^THx_k + c^Tx_k] \end{aligned}$$

in this problem. Simplifying $\frac{df(x+\alpha p)}{d\alpha} = 0$ gives

$$\begin{cases} a_1\alpha_k^3 + a_2\alpha_k^2 + a_3\alpha_k + a_4 = 0, \text{ where} \\ a_1 = 4\gamma(p_k^Tp_k)^2, \\ a_2 = 12\gamma(p_k^Tp_k)(x_k^Tp_k), \\ a_3 = 4\gamma((x_k^Tx_k)(p_k^Tp_k) + 2(x_k^Tp_k)^2) + p_k^THp_k, \\ a_4 = 4\gamma(x_k^Tx_k)(x_k^Tp_k) + (x_k^THp_k + c^Tp_k). \end{cases}$$

The real α_k solving the equation above performs an exact line search for a given point x_k and search direction p_k on the stated objective function $f(x) = \gamma(x^Tx)^2 + \frac{1}{2}x^THx + c^Tx$.

Appendix D: Algorithm to generate the convex random Q matrix

The random Q matrix for Testproblem 3 is generated according to Algorithm 8.3. Within the algorithm to be presented, four Matlabfunctions are being used. These will be described briefly below. The first function is *rand(m)*, which generate a $m \times m$ matrix with random entries from a uniform distribution on the interval (0,1). The second function is *mod(Q,1)*, which gives the modulus of Q after the division with 1. The third and fourth functions are used together in the command *min(eig(Q))*.

The *eig(Q)* function gives a vector containing the eigenvalues of the square matrix Q. The *min(eig(Q))* function returns the smallest element of the vector *eig(Q)*. The algorithm which generate a symmetric positive definite random matrix Q are presented below.

Algorithm 8.3. *Generate Q matrix algorithm*

$m \leftarrow$ size of wanted square matrix;
$Q \leftarrow 10\, rand(m)$;
$Q \leftarrow Q - mod(Q, 1)$;
$Q \leftarrow Q' + Q$;
if $min(eig(Q)) \leq 0$ **then**
    $k \leftarrow$ the smallest integer fulfilling $> |min(eig(Q))|$;
    $Q \leftarrow Q + kI$;
end

References

[1] M. Al-Baali. Global and superlinear convergence of a restricted class of self-scaling methods with inexact line searches, for convex functions. *Comput. Optim. Appl.*, 9(2):191–203, 1998.

[2] M. Al-Baali. Numerical experience with a class of self-scaling quasi-Newton algorithms. *J. Optim. Theory Appl.*, 96(3):533–553, 1998.

[3] R. H. Byrd, J. Nocedal, and Y. X. Yuan. Global convergence of a class of quasi-Newton methods on convex problems. *SIAM J. Numer. Anal.*, 24(5):1171–1190, 1987.

[4] F. Carlsson and A. Forsgren. Iterative regularization in intensity-modulated radiation therapy optimization. *Medical Physics*, 33(1):225–234, 2006.

[5] W. C. Davidon. Variable metric methods for minimization. *Argonne National Lab Report (Argonne, IL)*, 1959.

[6] J. W. Denton and M. S. Hung. A comparison of nonlinear optimization methods for supervised learning in multilayer feedforward neural networks. *European Journal of Operational Research*, 93:358–368, 1996.

[7] L. C. W. Dixon. Quasi-Newton algorithms generate identical points. *Math. Programming*, 2:383–387, 1972.

[8] R. Fletcher. *Practical Methods of Optimization*. Second ed., John Wiley & Sons, Chichester, 1987.

[9] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Comput. J.*, 7:149–154, 1964.

[10] P. E. Gill and M. W. Leonard. Limited-memory reduced-Hessian methods for large-scale unconstrained optimization. *SIAM J. Optim.*, 14(2):380–401 (electronic), 2003.

[11] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, 2003.

[12] G. H. Golub and C. F. Van Loan. *Matrix Computations*. North Oxford Academic, Oxford, 1983.

[13] E. Haber. Quasi-Newton methods for large-scale electromagnetic inverse problems. *Inverse Problems*, 21(1):305–333, 2005.

[14] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards*, 49:409–436 (1953), 1952.

[15] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, 1996.

[16] A. D. Klose and A. H. Hielscher. Quasi-Newton methods in optical tomographic image reconstruction. *Inverse Problems*, 19(2):387–409, 2003.

[17] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Research Nat. Bur. Standards*, 45:255–282, 1950.

[18] D. G. Luenberger. *Linear and Nonlinear Programming*. Second ed., Addison-Wesley, New York, 1989.

[19] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, New York, 1996.

[20] L. Nazareth. A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms. *SIAM J. Numer. Anal.*, 16(5):794–800, 1979.

[21] J. Nocedal. Theory of algorithms for unconstrained optimization. In *Acta numerica, 1992*, Acta Numer., pages 199–242. Cambridge Univ. Press, Cambridge, 1992.

[22] J. Nocedal. *Numerical Optimization*. Springer, New York, 1999.

[23] J. Nocedal and Y. X. Yuan. Analysis of a self-scaling quasi-Newton method. *Math. Programming*, 61(1, Ser. A):19–37, 1993.

[24] S. S. Oren. Self-scaling variable metric algorithms for unconstrained minimization. *Ph.D. Thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, California*, 1972.

[25] S. S. Oren. Self-scaling variable metric (SSVM) algorithms. II. Implementation and experiments. *Management Sci.*, 20:863–874, 1973/74. Mathematical programming.

[26] S. S. Oren and D. G. Luenberger. Self-scaling variable metric (SSVM) algorithms. I. Criteria and sufficient conditions for scaling a class of algorithms. *Management Sci.*, 20:845–862, 1973/74. Mathematical programming.

[27] A. P. Plumb, R. C. Rowe, P. York, and M. Brown. Optimisation of the predictive ability of artificial neural network (ANN) models: A comparison of three ANN programs and four classes of training algorithm. *European Journal of Pharmaceutical Sciences*, 25:395–405, 2005.

[28] M. J. D. Powell. *Some global convergence properties of a variable metric algorithm for minimization without exact line searches.* in SIAM-AMS Proceedings, Volume 9, Eds (R. W. Cottle and C. E. Lemke), SIAM Publications, Philadelphia, 1976.

[29] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Second ed., Society for Industrial and Applied Mathematics, Philadelphia, 2003.